Rotas com Express

PROGRAMAÇÃO WEB 1

Objetivos de aprendizagem

- Aprimorar o uso de rotas com Node.JS
- Conhecer as funcionalidades da estrutura de diretórios gerada

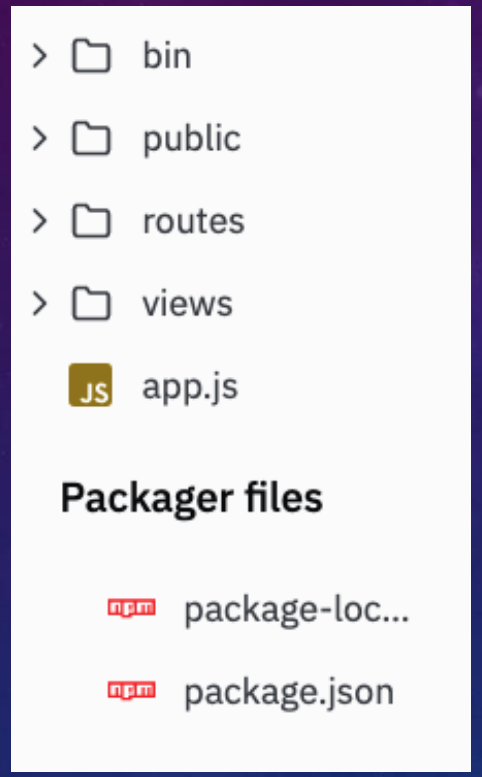
Agenda

- Criando um projeto com `express-generator`
- Estrutura básica
- Servindo arquivos estáticos
- Rotas
- *Templates*

Express generator

1. Criar um novo projeto em https://replit.com ou com a IDE de preferência
2. Deletar o arquivo index.js
3. No console, digitar:
    1. `npm install express express-generator nodemon`
    2. `npx express-generator`
    3. `npm install`
4. Usar o `nodemon` (opcional)
    1. `npx nodemon`

Estrutura padrão



- Ao final da execução dos comandos devemos ter a estrutra mostrada ao lado

Executando a aplicação

- Para executar a API:
  - `npm start`

NÃO UTILIZAR O BOTÃO "RUN"

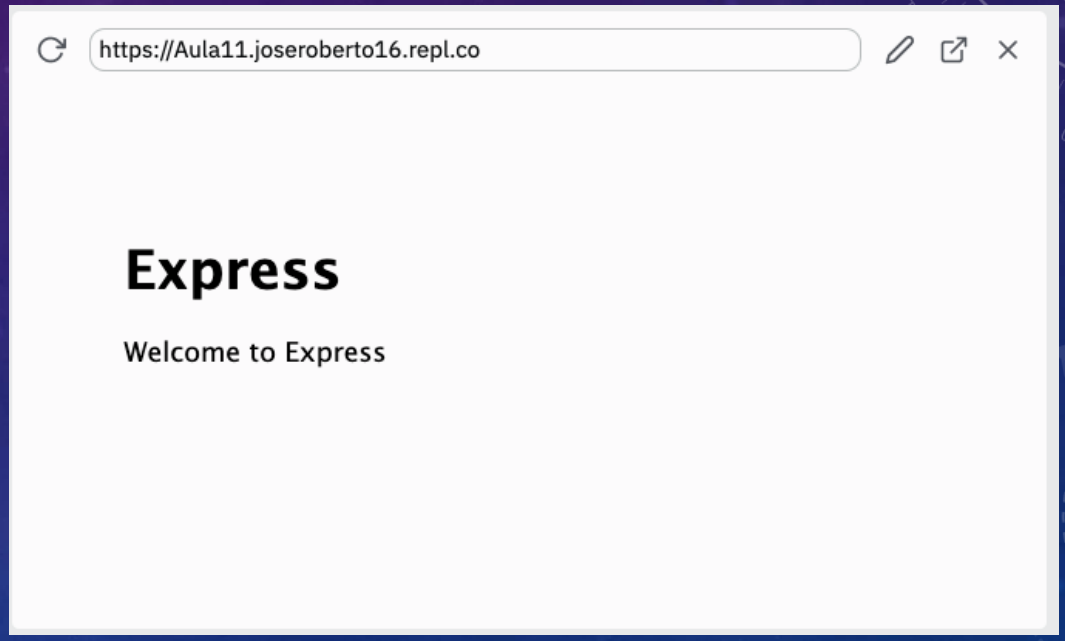
package.json

```
{
  "name": "aula11",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
    "morgan": "~1.9.1"
  }
}

```

Executando a aplicação

- Ao lado um *screenshot* da aplicação padrão QUE deve ser mostrada



package.json

- Define as dependências da aplicação
- Informações básicas do projeto
  - Nome
  - Versão
  - Autor
- *Script* de inicialização
  - ./bin/www

package.json

```
{
  "name": "aula11",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
    "morgan": "~1.9.1"
  }
}
```

/bin/www

- Ponto de entrada da API
- *Script* que inicializa a aplicação (node ./bin/www)
- Ao utilizar npm start será inicilizada com o comando apontado em “start”
- Inicializa o servidor web
- Ajusta a porta padrão (3000)
- Carrega o verdadeiro ponto de entrada da API (app.js)

app.js

- Cria um objeto aplicação express
- Configura a aplicação e seus *middlewares*
- Exporta a aplicação

```
var express =
require('express');
var app = express();
...
module.exports = app;
```

/public

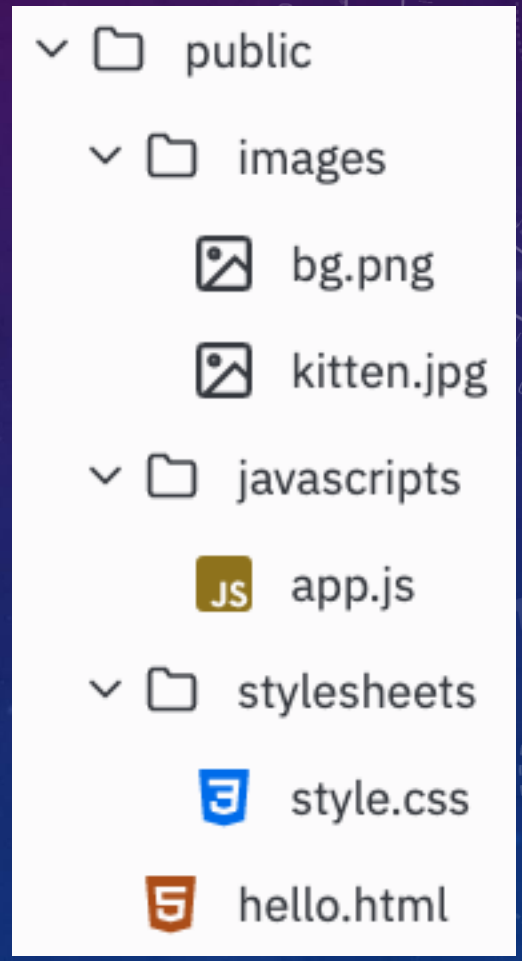
- Módulo `express.static`
- Para disponibilizar arquivos estáticos como imagens, CSS e JS utiliza-se, por exemplo:
  - `app.use(express.static(`public`)`
  - Disponibiliza os arquivos em public

Arquivos estáticos

```
app.use(express.static(`public`)
```

- Disponibiliza os arquivos de `/public`
- O nome do diretório não faz parte do caminho

```
http://localhost:3000/images/kitten.jpg
http://localhost:3000/css/style.css
http://localhost:3000/js/app.js
http://localhost:3000/images/bg.png
http://localhost:3000/hello.html
```

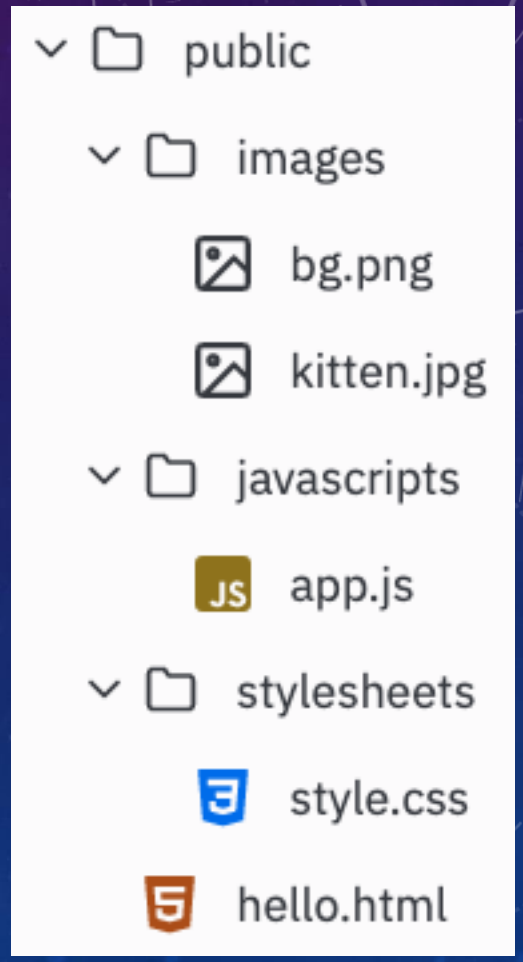


Arquivos estáticos

```
app.use('/static', express.static('public'))
```

- Cria um caminho virtual para os arquivos de /public

```
http://localhost:3000/static/images/kitten.jpg
http://localhost:3000/static/css/style.css
http://localhost:3000/static/js/app.js
http://localhost:3000/static/images/bg.png
http://localhost:3000/static/hello.html
```



/routes

- No diretório do exemplo temos dois arquivos:
  - index.js
  - users.js
- Basicamente possuem a mesma estrutura
- Todos os caminhos se referem a /

routes/index.js

```
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET home page. */
5  router.get('/', function(req, res, next) {
6    res.render('index', { title: 'Express' });
7  });
8
9  module.exports = router;
```

/routes

- Em **app.js** as rotas são especificadas
- Em **app.js** observar as linhas 7 e 22 para rota /
- E as linhas 8 e 23 para a rota /users

app.js ×

```
6
7  var indexRouter = require('./routes/index');
8  var usersRouter = require('./routes/users');
9
10 var app = express();
```

app.js ×

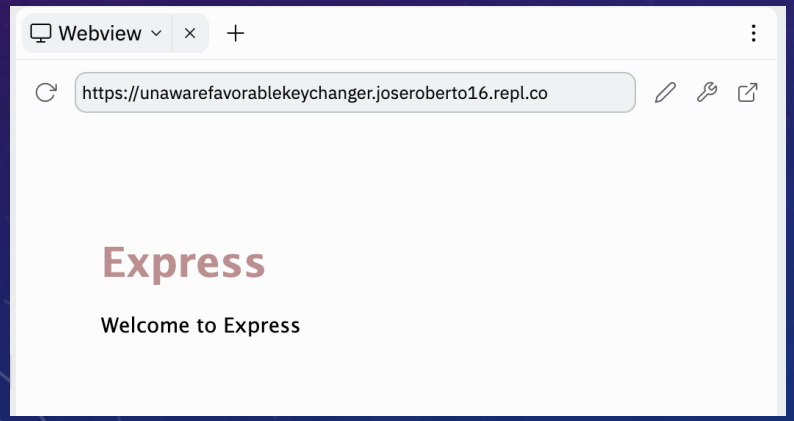
```
21
22 app.use('/', indexRouter);
23 app.use('/users', usersRouter);
24
```

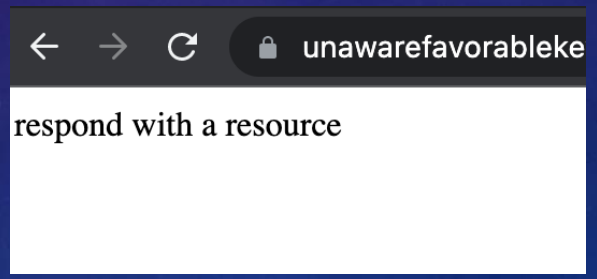
/routes

- Para manter a aplicação organizada o arquivo **app.js** apenas importa cada arquivo de rotas contido em `/routes`
- E entradas **app.use** de cada uma das rotas

O que acontece se....

1. Inicie a aplicação com `npm start` (Não usar o botão "Run")
2. A aplicação possui duas rotas (ver linhas 22 e 23 do `app.js`)
3. Acessar ambas as rotas. Abaixo o *screenshot* do que deve ser exibido





O que acontece se....

1. Observar que no diretório `/routes` existem dois arquivos que são apontados como arquivos de rota nas linhas 7 e 8 do `app.js`
2. Abrir o arquivo `index.js`
3. Observar que são adicionados os módulos `express` (linha 1) e que é instanciado um router (linha 2)
4. Ao invés de utilizar `app.get` ou similar, utiliza-se `router.get`
5. Observar a função `res.render` que renderiza a página com o *template* "index" (`/views`)
6. Na linha 9 é feita a exportação do router criado na linha 2

routes > index.js > ...

```
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET home page. */
5  router.get('/', function(req, res, next) {
6    res.render('index', { title: 'Express' });
7  });
8
9  module.exports = router;
```

`res.render()`

- Em `index.js` observar a linha 6
- As requisições são renderizadas devolvendo ao cliente uma página HTML, conforme o template apontado (index)

```
5 ▼ router.get('/', function(req, res, next) {
6     res.render('index', { title: 'Express' });
7   });
```

/views

- Armazena os *templates* utilizados no Express
- Existem diversos *template engines* (pug, mustache, EJS, etc)
- O padrão é Jade (.jade) que foi atualizado e agora chama-se Pug (.pug)
- Para utilizar um *template engine* é necessário configurar dois parâmetros:
  - Diretório
  - *Template Engine*

/views

Diretório

- Por exemplo,

```
app.set('views',
'./views’)
```

- Observar app.js (linha 13)

```
app.set('views',
path.join(__dirname,
'views'));
```

Template Engine

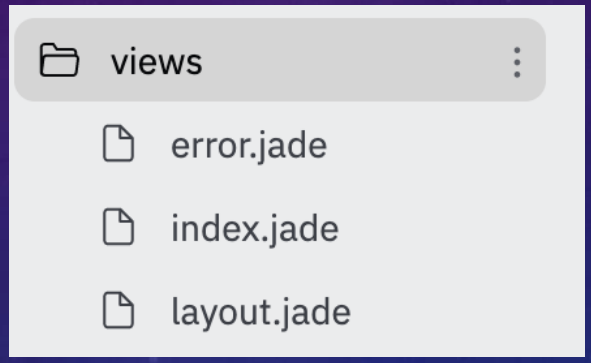
- Por exemplo,

```
app.set('view engine',
'jade')
```

- Para usar o PUG

```
app.set('view engine',
'pug')
```

/views



views > index.jade

```
1  extends layout
2
3  block content
4    h1= title
5    p Welcome to #{title}
6
```

views > layout.jade

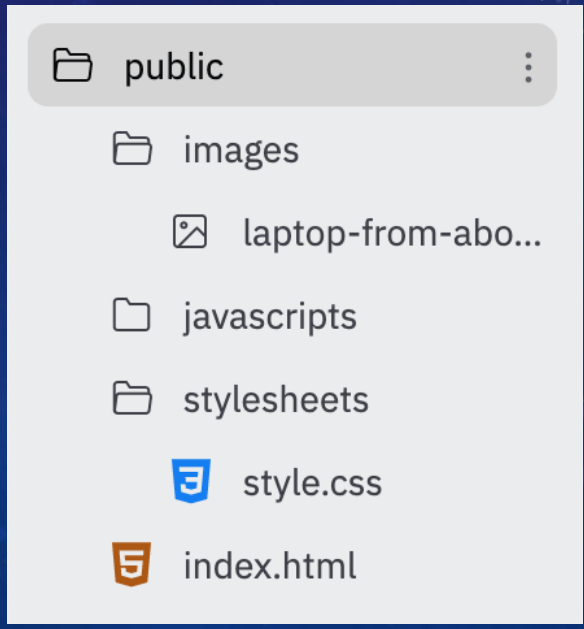
```
1  doctype html
2  html
3    head
4      title= title
5      link(rel='stylesheet', href='/stylesheets/style.css')
6    body
7      block content
8
```

Perguntas / Dúvidas

Exercício 1

Utilizando o `express` em conjunto com `express-generator`, explore as possibilidades da utilização de um router do express

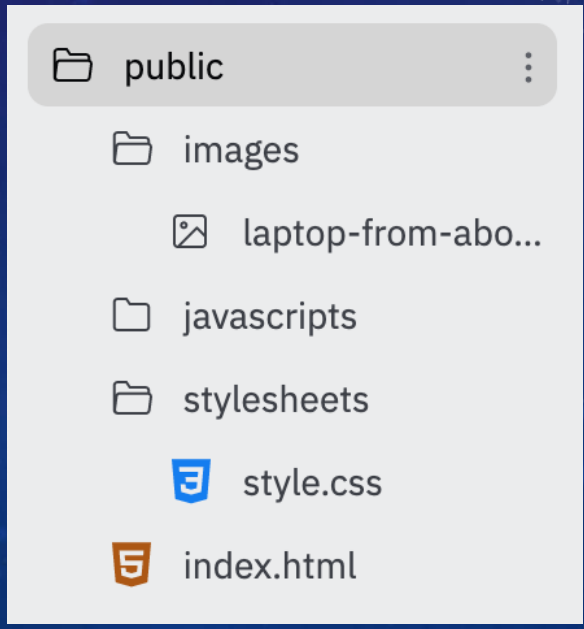
1. Criar um novo projeto nodejs no Replit;
2. Apagar o `index.js` existente
3. Instalar os módulos necessários como npm (`express` e `express-generator`)
4. Iniciar a aplicação
5. Observar que já existem duas rotas: / e /users
6. Acrescentar as rotas: /news e /about. Faça os devidos ajustes no arquivo app.js
7. Criar os respectivos arquivos de rotas no diretório /routes (news.js e about.js). Replicar o conteúdo do arquivo users.js em news.js e about.js
8. Modificar a mensagem "respond with a resource" para uma mensagem diferente que identifique a rota que está sendo acessada
9. Teste as rotas criadas
10. Modificar a variável `title` no arquivo `index.js`. O que acontece?
11. Teste novamente a aplicação



Exercício 2

Utilizando o `express` em conjunto com `express-generator`, crie um mini site estático.

1. Criar um novo projeto nodejs no Replit;
2. Apagar o `index.js` existente
3. Instalar os módulos necessários como npm (`express` e `express-generator`)
4. Utilizar os arquivos os arquivos disponibilizados na aula 13 para criar a estrutura mostrada ao lado
5. Rodar a aplicação com `npm start` (Não utilizar o botão "Run")
6. O que acontece? As figuras foram carregadas? A folha de estilo foi carregada?
7. Modificar o arquivo `index.html` para que a folha de estilo e a figura sejam carregadas.
8. Adicionar uma segunda página `mybedroom.html` similar a página `index.html`



Referências

- https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/skeleton_website
- https://expressjs.com/en/guide/routing.html
- https://expressjs.com/en/starter/static-files.html
- https://expressjs.com/en/guide/using-template-engines.html